LEARNING DECISION TREES AS AMORTIZED STRUCTURE INFERENCE

Mohammed Mahfoud[1,2]*, **Ghait Boukachab**[1], **Michał Koziarski**[3,4], **Alex Hernandez-Garcia**[1,5], **Stefan Bauer**[2], **Yoshua Bengio**[1,5], **Nikolay Malkin**[6]

[1]Mila – Québec AI Institute
[2]Technical University of Munich
[3]The Hospital for Sick Children
[4]Vector Institute
[5]Université de Montréal
[6]University of Edinburgh

ABSTRACT

Building predictive models for tabular data presents fundamental challenges, notably in scaling consistently, *i.e.*, more resources translating to better performance, and generalizing systematically beyond the training data distribution. Designing decision tree models remains especially challenging given the intractably large search space, and most existing methods rely on greedy heuristics, while deep learning inductive biases expect a temporal or spatial structure not naturally present in tabular data. We propose a hybrid *amortized structure inference* approach to learn predictive decision tree ensembles given data, formulating decision tree construction as a *sequential planning* problem. We train a deep reinforcement learning (GFlowNet) policy to solve this problem, yielding a generative model that samples decision trees from the Bayesian posterior. We show that our approach, DT-GFN, outperforms state-of-the-art decision tree and deep learning methods on standard classification benchmarks derived from real-world data, robustness to distribution shifts, and anomaly detection, all while yielding interpretable models with shorter description lengths. Samples from the trained DT-GFN model can be ensembled to construct a random forest, and we further show that the performance of scales consistently in ensemble size, yielding ensembles of predictors that continue to generalize systematically. Code available at: https://github.com/GFNOrg/dt-gfn.

1 INTRODUCTION

Tabular data is a common modality across a variety of fields where machine learning is employed, *e.g.*, healthcare (Przystalski & Thanki, 2024) and finance (Dixon et al., 2020). Unlike data with temporal, spatial or graph structure—such as text, images and molecules, where deep learning methods have seen their most visible successes (OpenAI, 2024; Anthropic, 2024; Gupta et al., 2024)—tabular data simply has the form of samples (rows) with a shared set of features (columns). Two widely adopted sets of machine learning methods that model the dependence of a target variable on the features are (i) methods based on rule learning, in particular, decision trees and their extensions; (ii) deep learning methods. Because deep learning has been less successful as general-purpose learner for tabular data than for domains where natural inductive biases for modeling exist, debate persists about the best way to build predictive models for this modality, whether it is deep neural networks (McElfresh et al., 2023; Holzmüller et al., 2024), gradient-boosted trees (Grinsztajn et al., 2022; Shwartz-Ziv & Armon, 2022), or others.

The key desideratum in designing consistently better models for tabular data is the ability to generalize systematically beyond the training distribution, a critical problem in machine learning (Hand, 2006; Quiñonero-Candela et al., 2022). Benchmarks like TABLESHIFT (Gardner et al., 2023) and WILDS (Koh et al., 2021) evaluate models' sensitivity to *distribution shifts* in tabular data, while ODDS (Rayana, 2016) and TABMEDOOD (Azizmalayeri et al., 2023) are used to study the related

*Corresponding author: mo.mahfoud@tum.de

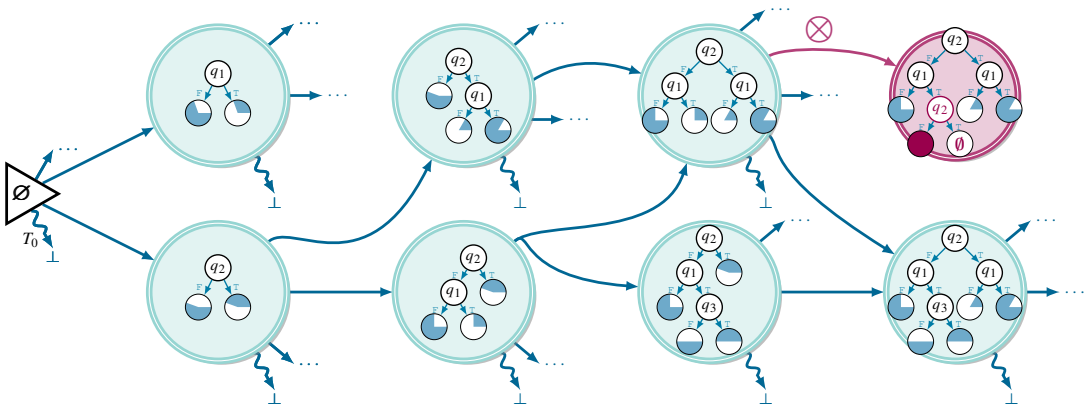

Figure 1: **Learning a decision tree as decision-making in a Markov decision process (MDP)** $\mathcal{M}$. At each step of construction, the data is split by a decision threshold on one of the features (right for True [T] or left for False [F]). We start from an empty source state T_0 with no decision rules and move through $\mathcal{M}$ by taking some action a corresponding to finding a decision rule (ℓ, f, t), *i.e.*, split data at leaf ℓ on feature f with threshold t. At each state of $\mathcal{M}$, a can either be a valid action, *i.e.*, resulting in a valid split, or invalid one, resulting in an invalid/redundant split. At each state of $\mathcal{M}$, we have the choice to stop sampling, in which case the resulting tree is a terminating state $\perp$. The reward function $\mathcal{R}$ can be computed at any valid state.

problem of *out-of-distribution detection*. Additional desiderata are *consistent scalability*—more resources in training or inference should lead to better model performance—and the ability to learn *reusable* knowledge and efficiently adapt to streaming data in online learning.

In this work, we treat the problem of learning decision trees from tabular data as a *structure inference* problem and propose to tackle it with deep reinforcement learning (RL) methods. Namely, we model the construction of a decision tree as a sequential decision-making process and learn a policy that constructs trees so as to sample from the Bayesian posterior distribution over decision tree models given data (Fig. 1). The policy is a deep neural network with inductive biases that take advantage of compositional structure in the target distribution over decision trees. Samples from the learned policy—a generative model over decision trees—can be used to construct ensembles of decision trees that are more expressive than individual trees while being probabilistically principled as Bayesian model averages. In our approach, which we call DT-GFN, deep neural networks are not used as predictors of target variables given features. Rather, they are used as *amortized inference models* in the construction of a rule-based model that acts as the predictor. Among other advantages, this enables the performance of the predictor to scale consistently in the number of tree models in the ensemble, while the individual trees remain short in data description length and interpretable. In summary, our main contributions are:

- We propose a sequential decision-making formulation for learning decision trees, leveraging deep RL methods (GFlowNets) to amortize the intractable sampling from the Bayesian posterior over decision trees.
- We evaluate trees, and Bayesian ensembles of trees, sampled from the learned generative models on standard tabular data benchmarks.
- Our approach compares favorably to state-of-the-art in standard classification tasks. We show improved generalization abilities in robustness to distribution shift and out-of-distribution tasks, while our tree samples remain comparatively shorter and more interpretable.
- We show evidence of consistent scaling ability in the ensemble size, *i.e.*, sampling more tree models yields better ensemble prediction; and in compute budget, allowing to adjust the performance-cost tradeoff as needed.

2 Related Work

Our work closely relates to various approaches to learning decision trees- with explicit *splitting* criteria, as Bayesian inference, and as dynamic programming and/or explicit optimization- along with tabular deep learning and aspects of amortized inference and GFlowNets. We provide a more detailed account of that in Appendix B.

3 Setting and Preliminaries

Notation. We denote the set containing the first n positive integers $\{1, 2, ..n\}$ as $[n]$, where $n \in \mathbb{N}$. $\mathbb{R}^+$ is the set of non-negative real numbers. We denote scalars in regular font, x, and vectors in bold, $\boldsymbol{x}$. For any set $\mathcal{S}$, $|\mathcal{S}|$ denotes the cardinality of $\mathcal{S}$. $\mathbb{I}$ denotes the indicator function, *i.e.*, $\mathbb{I}^{\mathcal{X}}(x) = 1$ if $x \in \mathcal{X}$ for some set $\mathcal{X}$ and 0 otherwise.

Setting. We consider a supervised learning setting where we have access to a labeled tabular dataset $\mathcal{D} = (\mathcal{X}, \mathcal{Y})$. $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^n$ is a set of features, where $\mathbf{x}_i \in \mathbb{R}^d$ and $\mathcal{Y} = \{y_i\}_{i=1}^n$ is a set of labels, where $y_i \in [C]$ with C denoting the number of distinct classes and $C = [C]$. We further assume that all data points $(\mathbf{x}_i, y_i)$ are i.i.d.

3.1 Bayesian Posterior over Decision Trees

An instance of a Bayesian decision tree (BDT) (T, Θ) partitions the feature space into a set of leaves $\mathcal{L} = \{\ell_1, \ldots, \ell_{|\mathcal{L}|}\}$. Each leaf ℓ is associated with a probability vector $\boldsymbol{\theta}_\ell = (\theta_{\ell,1}, \theta_{\ell,2}, \ldots, \theta_{\ell,C})$, yielding $\Theta = \{\boldsymbol{\theta}_{\ell_1}, \ldots, \boldsymbol{\theta}_{\ell_{|\mathcal{L}|}}\}$. For each leaf ℓ, we define the partitioning function $\Delta(\ell)$ which partitions the sample subsets of points $\Delta(\ell_1), \ldots, \Delta(\ell_{|\mathcal{L}|})$ falling under ℓ.

Proposition 3.1 (Likelihood of a Bayesian DT)**.** *The likelihood under a Bayesian decision tree (T, Θ) given features $\mathcal{X}$ and labels $\mathcal{Y}$ is written as follows*

$$\mathbb{P}\left[\mathcal{Y}|\mathcal{X}, T, \Theta\right] = \prod_{\ell \in \mathcal{L}} \prod_{i \in \Delta(\ell)} \prod_{c \in C} \theta_{\ell,(c)}^{\mathbb{I}^{(c)}(y_i)} = \prod_{\ell \in \mathcal{L}} \prod_{c \in C} \theta_{\ell,c}^{\sum_{i \in \Delta(\ell)} \mathbb{I}^{(c)}(y_i)},$$

where $\mathcal{L}$ is the set of leaves in the tree T, C is the set of classes, $\theta_{\ell,(c)}$ is the probability of sampling class c under leaf ℓ, and $\mathbb{I}^{(c)}(y_i)$ is the indicator function of whether y_i belongs to class c.

The Bayesian Classification and Regression Trees (BCART) construction (Chipman et al., 1998) assumes a Dirichlet prior distribution over Θ, *i.e.*, $\boldsymbol{\theta} \sim \text{Dirichlet}(\boldsymbol{\alpha})$ for all $\boldsymbol{\theta} \in \Theta$. Under this prior, we express the marginal likelihood $\mathbb{P}\left[\mathcal{Y}|\mathcal{X}, T\right]$ as follows.

Proposition 3.2 (Marginal Likelihood of a Bayesian DT)**.** *Assuming $\boldsymbol{\theta} \sim Dirichlet(\boldsymbol{\alpha})$, the marginal likelihood of a Bayesian decision tree T given features $\mathcal{X}$ and labels $\mathcal{Y}$*

$$\mathbb{P}\left[\mathcal{Y}|\mathcal{X}, T\right] = \left(\frac{\Gamma(\sum_{c \in C} \alpha_c)}{\prod_{c \in C} \Gamma(\alpha_c)}\right)^{|\mathcal{L}|} \prod_{\ell \in \mathcal{L}} \frac{\prod_{c \in C} \Gamma(n_{\ell,c} + \alpha_c)}{\Gamma(n_\ell + \sum_{c \in C} \alpha_c)},$$

where $n_{\ell,c} = \sum_{i \in \Delta(\ell)} \mathbb{I}^{(c)}(y_i)$ is the empirical count of data points with label c at leaf ℓ and $n_\ell = \sum_{c \in C} n_{\ell,c}$ is the total empirical count of data points (of all classes) at leaf ℓ.

We defer the proofs of the aforementioned results along with a more in-depth disussion of the role or priors to Appendix D.1. As the prior over tree structures $\mathbb{P}\left[T|\mathcal{X}\right]$ remains a design choice, we defer discussing that to §4.2.

Decision tree search space size. The space of decision trees is enormous even for small depths and numbers of features. According to Hu et al. (2019), assuming a (full) binary tree of depth d_t and given a dataset with p binary features, the number of distinct trees is

$$N_{d_t} = \sum_{n_0=1}^{1} \sum_{n_1=1}^{2n_0} \cdots \sum_{n_{d_t-1}=1}^{2n_{d_t-2}} p \times \binom{2n_0}{n_1} (p-1)^{n_1} \times \cdots \quad \times \binom{2n_{d_t-2}}{n_{d_t-1}} (p - (d_t - 1))^{n_{d_t-1}}. \tag{1}$$

Hence, the size of the search space over decision trees up to some depth d is simply $\sum_{d_t=1}^{d} N_{d_t}$. §3.1 computes the latter for $d \in \{1, 2, 3, 4, 5\}$ given a dataset with $p \in \{10, 20\}$ binary features; exact computation for $d = 5$ is already prohibitive, at least by elementary means.

d	$p = 10$	$p = 20$
1	1.000×10^1	2.000×10^1
2	1.000×10^3	8.000×10^3
3	5.329×10^6	9.411×10^8
4	5.609×10^{13}	8.358×10^{18}

Given this challenge, we propose to use reinforcement learning methods to amortize search over this very large space, and the benefits of GFlowNets as amortized, diversity-seeking samplers.

3.2 Amortized Inference with GFlowNets

We briefly recall some of the main ideas behind GFlowNets relevant to our context, along with key ingredients to train them. A GFlowNet assumes access to a fully observable deterministic MDP with a set of states $\mathcal{S}$ and set of actions $\mathcal{A} \subseteq \mathcal{S} \times \mathcal{S}$. From the states in $\mathcal{S}$, we note in particular that the MDP $\mathcal{M}$ has a unique *source state* s_0 with no parents (represented as T_0 in Fig. 1), and a subset of states $\mathcal{X} \subset \mathcal{S}$ which we refer to as *terminal states* (represented as $\perp$ in Fig. 1); terminal states have no outgoing actions and subsequently no children states. We assume that any state in $\mathcal{S}$ is reachable from s_0 through some sequence of actions as illustrated in Fig. 1. We refer to a sequence of states $(\mathbf{s}_i \to \cdots \to \mathbf{s}_j)$ as a *trajectory*, such that a transition between each pair of consecutive states $(\mathbf{s}_t \to \mathbf{s}_{t+1})$ is induced by an action $a \in \mathcal{A}$. In particular, we define a *complete trajectory* as a trajectory that starts from s_0 and ends at some $s_n \in \mathcal{X}$, *i.e.*, $\tau = (\mathbf{s}_0 \to \mathbf{s}_1 \to \cdots \to \mathbf{s}_n = \mathbf{x})$.

Policy. A *(forward) policy* operating on the aforementioned MDP outputs a distribution $\mathbb{P}_F[\mathbf{s}'|\mathbf{s}]$ for each state $\mathbf{s} \in \mathcal{S} \setminus \mathcal{X}$ over the states $\mathbf{s}'$ that are reachable from $\mathbf{s}$ within a single action[1]. Given a complete trajectory τ, a policy induces a distribution over τ written as

$$\mathbb{P}_F\left[\tau = (\mathbf{s}_0 \to \mathbf{s}_1 \to \cdots \to \mathbf{s}_n = \mathbf{x})\right] = \prod_{t=0}^{n-1} \mathbb{P}_F\left[\mathbf{s}_{t+1}|\mathbf{s}_t\right]. \tag{2}$$

We denote the marginal distribution over terminal states by $\mathbb{P}_F^\top$. In general, estimating $\mathbb{P}_F^\top$ exactly or computing it in closed-form is intractable

$$\mathbb{P}_F^\top[\mathbf{x}] = \textstyle\sum_{\tau \rightsquigarrow \mathbf{x}} \mathbb{P}_F^\top[\tau], \tag{3}$$

where the sum is taken over all complete trajectories leading to $\mathbf{x}$.

Inheriting terminology from the RL literature, a *reward function* $\mathcal{R}(\mathbf{x})$ for a GFlowNet represents the target—possibly and typically unnormalized—distribution over the set of terminal states $\mathcal{X}$. Formally, $\mathcal{R}$ is a mapping from the set of terminal states $\mathcal{X}$ to $\mathbb{R}^+$, *i.e.*, $\mathcal{R} : \mathcal{X} \to \mathbb{R}^+$. A typical form the reward function takes in our construction is $\mathcal{R}(\mathbf{x}) = e^{-\mathcal{E}(\mathbf{x})/T}$ where $\mathcal{E} : \mathcal{X} \to \mathbb{R}$ is an energy function and $T \in \mathbb{R}^+$ is a temperature control parameter. Given the latter, the learning problem a GFlowNet aims to approximate is sequentially choosing a policy $\mathbb{P}_F[s_{t+1}|s_t]$ such that the induced marginal distribution $\mathbb{P}_F^\top[s_n = x]$ is proportional to the reward function evaluated at $\mathbf{x}$ up to a normalizing constant, that is

$$\mathbb{P}_F^\top[x] \propto \mathcal{R}(\mathbf{x}) = e^{-\mathcal{E}(\mathbf{x})/T}. \tag{4}$$

$\mathbb{P}_F$ is typically parametrized by a neural network with parameters $\boldsymbol{\theta}$, denoted $\mathbb{P}_F[\bullet\,;\,\boldsymbol{\theta}]$, that outputs the distribution over states $\mathbf{s}'$ reachable from any state $\mathbf{s} \in \mathcal{S}$ when taking argument $\mathbf{s}'|\mathbf{s}$.

A key challenge within this line of framing is the potential intractability of the number of trajectories leading to some object $\mathbf{x}$, especially when the search space becomes combinatorially large and trajectories become longer. This makes $\mathbb{P}_F^\top$ and the normalizing constant in (4), $Z = \sum_{\tau \rightsquigarrow \mathbf{x}} \mathcal{R}(\mathbf{x})$, hard to estimate. Various objectives have been proposed for circumventing this, usually via injecting additional learnable objects or parameters into the optimization problem. In particular, we use *trajectory balance* (Malkin et al., 2022), and introduce it in the following.

Trajectory balance (TB). The TB objective is constructed by introducing an additional *backward policy* $\mathbb{P}_B$, which can be learned or fixed, and a single scalar $Z_{\boldsymbol{\theta}}$ that estimates the partition function on the reward right hand side of (4), *i.e.*, $Z = \sum_{\tau \rightsquigarrow \mathbf{x}} \mathcal{R}(\mathbf{x})$. More precisely, $\mathbb{P}_B$ is defined as a collection of distributions $\mathbb{P}_B[\bullet|\mathbf{s}]$ over the parent states of any non-source state $\mathbf{s}$ and induces a distribution over complete trajectories τ leading to $\mathbf{x}$, *i.e.*,

$$\mathbb{P}_B\Big[\tau = (\mathbf{s}_0 \leftarrow \mathbf{s}_1 \leftarrow \cdots \leftarrow \mathbf{s}_n = \mathbf{x})|\mathbf{x}\Big] = \prod_{t=0}^{n-1} \mathbb{P}_B\Big[\mathbf{s}_t|\mathbf{s}_{t+1}\Big]. \tag{5}$$

Via this parametrization, the TB objective reduces enforcing (4) to (more simply) enforcing $\mathbb{P}_F[\tau] \propto \mathbb{P}_B[\tau|\mathbf{x}] \cdot \mathcal{R}(\mathbf{x})$ for every complete trajectory τ ending in $\mathbf{x}$ (note that the latter implies the former). Such proportionality is directly enforced by the training loss:

$$\ell_{\text{TB}}(\tau;\boldsymbol{\theta}) = \left(\log\Big[Z_{\boldsymbol{\theta}} \cdot \mathbb{P}_F[\tau;\boldsymbol{\theta}]\Big] - \log\Big[\mathbb{P}_B[\tau|\mathbf{x};\boldsymbol{\theta}] \cdot \mathcal{R}(\mathbf{x})\Big]\right)^2. \tag{6}$$

[1]Note here that we can write the policy as a distribution over "next" states (from current state) or actions interchangeably as the MDP is deterministic.

If $\ell_{\text{TB}}(\tau;\boldsymbol{\theta}) = 0$ for all complete trajectories τ, then the output policy $\hat{\mathbb{P}}_F$ provably samples proportionally to the reward function, *i.e.*, (4), and the output normalization constant Z_θ equals the partition function of the reward, *i.e.*, $Z_\theta = \sum_{\tau \rightsquigarrow \mathbf{x}} \mathcal{R}(\mathbf{x})$ (Malkin et al., 2022). Another important resulting observation is that given some fixed $\mathbb{P}_B$, there exists a unique $\hat{\mathbb{P}}_F$ that satisfies (4), which allows to set $\mathbb{P}_B$ to some fixed distribution at initialization.

Exploration in training. The TB objective operates on trajectories in training, yet the process of choosing the trajectories to optimize (6) remains an algorithmic choice. A default choice is to train *on-policy* by sampling $\tau \sim \mathbb{P}_F[\tau;\boldsymbol{\theta}]$ and minimizing $\ell_{\text{TB}}(\tau;\boldsymbol{\theta})$ with gradient descent. Given GFlowNets' aim to sample diversely from the reward function by design, favoring exploration in training has also proven to yield a variety of benefits. This could be done by sampling τ from a tempered $\mathbb{P}_F$ or through sampling individual actions from a uniform distribution ϵ of the time on average, akin to ϵ-greedy exploration in RL.

4 Learning Decision Trees as Amortized Structure Inference

4.1 Constructing the GFlowNet's underlying MDP $\mathcal{M}$

State. Given a choice of a maximum tree depth d_{max} a tree is allowed to expand to, there are a maximum of $2^{d_{\text{max}}+1} - 1$ nodes in T. We represent each node as a vector, which we order following a breadth-first traversal. Each node is either a decision node, in which case its corresponding vector is a decision rule, or a leaf node in which case it is labeled as an end-of-sentence (<EOS>). A decision rule is defined as a tuple (f_i, t_i), where $f_i \in [d]$ is some given feature and $t_i \in \mathbb{R}$ is some real number corresponding to the splitting threshold for f_i. By notation, data-points satisfying $f_i \leq t_i$ flow to the left child node, and the rest flow to the right one. A choice at a child node is only allowed if a choice of a decision rule was made for all parent nodes. At initialization, each entry is assigned a "not yet specified" value of $\varnothing$. At the root node, either a decision rule or termination are chosen; if a decision rule is picked, T is then recursively augmented in a similar way.

Action. We opt to keep the action space discrete for simplicity. For that, we scale features back to $[0, 1]$ and pick t thresholds uniformly spaced in this range; t remains a hyperparameter of choice. An alternative paradigm could be using quantiles of training data, which we have found to often overfit the train set. As both the number of possible thresholds f_i and t_i can be large, we define the action space hierarchically. At each leaf node of each non-terminal tree state T, we can either pick a decision rule or terminate. Picking a decision rule consists in hierarchically choosing a feature f_i, then a threshold t_i. To account for decisions resulting in invalid states (see Fig. 1), we mask thresholds that do not result in a valid split, *i.e.*, one where at least one data-point flows to each child node. A terminal state is reached either when an action to terminate is picked, or when all actions are masked.

4.2 Reward Function and Parameter Sampling

Reward. As outlined in §3, we choose our reward function to be the joint conditional probability distribution $\mathcal{R}(T|\mathcal{X}) = \mathbb{P}[\mathcal{Y}|\mathcal{X}, T] \cdot \mathbb{P}[T|\mathcal{X}] \propto \mathbb{P}[T|\mathcal{Y}, \mathcal{X}]$. Unlike a variety of settings where GFlowNets have been applied, the reward function can be written in closed form without any further parametrizations. The remaining ingredient is to choose a prior on the structure of T. Anchored in Occam's razor (Rissanen (1978); Chapter 28, MacKay (2003)) and akin to previous work on (provably) optimal decision trees (Hu et al., 2019; Lin et al., 2020; Balcan & Sharma, 2024), we choose a prior of minimal (data) description length of T, as measured by its number of decision nodes. To properly normalize the likelihood term, we further inject a parameter β, and provide guidelines on tuning it appropriately. We formulate the posterior distribution over Bayesian DTs given our chosen prior $\mathbb{P}[T|\mathcal{X}] = e^{-\beta \cdot n(T)}$ as follows

$$\mathbb{P}[T|\mathcal{Y}, \mathcal{X}] \propto \mathbb{P}[\mathcal{Y}|\mathcal{X}, T] \cdot \mathbb{P}[T|\mathcal{X}] \propto e^{-\beta \cdot n(T)} \cdot \left(\frac{\Gamma(\sum_{c \in C} \alpha_c)}{\prod_{c \in C} \Gamma(\alpha_c)} \right)^{|\mathcal{L}|} \prod_{\ell \in \mathcal{L}} \frac{\prod_{c \in C} \Gamma(n_{\ell,c} + \alpha_c)}{\Gamma(n_\ell + \sum_{c \in C} \alpha_c)}, \quad (7)$$

where $\beta \geq 0$ and $n(T)$ is the number of decision (non-leaf) nodes in a tree T.

Choice of β. An important consideration when computing the prior over trees is the choice of the parameter β, which controls how much we penalize overly complex trees. For a choice of a large β, our model might not be expressive enough. For a choice of a small β, our model might become overly complex, potentially undermining its interpretability. We would like to pick some β such that the resulting model would be expressive enough, have a short description length and be interpretable.

Relying on information theoretical arguments, as outlined in Appendix D.2, we propose that a suitable choice of β is $\beta \sim \log(4) + \log(d) + \log(t)$, where d is the number of splitting features, and t is the number of splitting thresholds corresponding to the chosen discretization of the feature support.

Sampling classification parameters at inference. Given a trained tree structure, classification parameters $\boldsymbol{\theta}_\ell$ at each leaf are sampled from the posterior $\boldsymbol{\theta}_\ell \sim \text{Dirichlet}(\boldsymbol{n}_\ell + \boldsymbol{\alpha})$, where $\boldsymbol{n}_\ell$ are the empirical counts of samples belonging to each class at leaf ℓ and $\boldsymbol{\alpha}$ is a prior on overall class counts.

4.3 Parametrization of the Forward Policy

A key property of our formulation is that it allows to frame decision tree learning as reasoning over root-to-leaf paths by picking a sequence of decision rules given a context of previous decisions. Given that both the prior and the predictions over different root-to-leaf paths are independent, a learned tree representation is invariant to their order.[2] A representation given as input to our policy model is a set of (padded) vectors, each representing the root-to-leaf path to a leaf. The termination probability $\mathbb{P}_F\left(\text{<EOS>}\middle|\mathbf{s}_t\right)$ is parametrized as a simple multi-layer perceptron (MLP). The likelihoods of other actions $\mathbb{P}_F\left(\mathbf{s}_{t+1}\middle|\mathbf{s}_t; \neg\text{<EOS>}\right)$ are parametrized by a second MLP, evaluated independently on the representation of each leaf. The backward policy $\mathbb{P}_B$ is simply set to be uniform over parent states. To encourage exploration and diversely sample from the posterior, trajectories need not always be sampled from $\mathbb{P}_F$ (*on-policy*). Instead, we use two widely adopted techniques from the RL literature, which have also been used for off-policy exploration in GFlowNets: a replay buffer and ϵ-random exploration with annealed ϵ. See Appendix F for all hyperparameters and training details.

5 Empirical Evaluation

In §5.1, we first evaluate our approach on two desirable properties when designing models for tabular data: 1) in-distribution generalization, as measured by held-out set accuracy, and 2) complexity (often also used as a proxy for interpretability), as measured by model size for DT-based methods, where model size directly correlates with model complexity. Next, in §5.2, we would like to evaluate the potential of constructing ensembles of predictors sampled from DT-GFN, and how these can be competitive with gradient-boosted trees and deep learning methods. In §5.3, we consider experiments in systematic generalization, where we would like to see how models learned via structure inference with our method compare to a variety of other baselines. Finally, we conduct consistent scaling experiments in §5.4. We further show in Appendix G.1 that DT-GFN compares favorably to state-of-the-art at minimum cost (see Fig. 4). For all experiments, we highlight the **best** result in bold and blue and most competitive (second best) result(s) in blue cell shades. For experiments in which we report model sizes, we omit mentioning the "best" result as it is hard to determine the optimal one. We further point to Appendix J for details on baseline implementations where needed.

5.1 Benchmarking with Single Decision Tree Algorithms

Table 1: **Test accuracy and model size (total number of tree nodes) for single decision tree baselines, averaged over five random seeds**. For each algorithm, to account for model variance, we construct 1000 trees and pick the best tree in training. For Bayesian algorithms (including our DT-GFN), we choose the tree with the highest log-posterior. For all of the others, we choose the tree with the highest accuracy in training.

Dataset →	**Iris**		**Wine**		**Breast Cancer**$^{(D)}$		**Raisin**	
Algorithm ↓	Test Acc↑	Size↓	Test Acc↑	Size↓	Test Acc↑	Size↓	Test Acc↑	Size↓
SMC	0.9518 ±0.02	16.18 ±1.72	0.9311 ±0.04	16.25 ±2.66	0.931 ±0.01	32.32 ±2.68	0.866 ±0.01	46.58 ±2.12
MCMC	0.923 ±0.04	13.4 ±1.5	0.955 ±0.02	13.82 ±1.21	0.92 ±0.02	25.62 ±2.67	0.864 ±0.02	35.29 ±1.93
MAPTREE	0.8733 ±0.04	3.80 ±0.45	0.9139 ±0.02	4.8±0.45	0.9281 ±0.02	5 ±0	0.8344 ±0.03	7.80 ±1.10
BCART	0.9267 ±0.03	56.2 ±31.8	0.9389 ±0.02	49.8 ±34.26	0.9018 ±0.03	20.6 ±12.09	0.8678 ±0.01	23.80 ±8.26
(α^*, β^*)-TSALLIS	0.9267 ±0.04	16.6 ±1.5	0.944 ±0.315	19 ±1.26	0.937 ±0.017	22.2 ±0.98	0.864 ±0.01	28.2 ±1.6
CART-GINI	0.9494 ±0.02	14.6 ±1.5	0.876 ±0.05	17.8 ±4.12	0.923 ±0.02	34.6 ±6.25	0.852 ±0.023	29.8 ±0.98
CART-ENTROPY	0.9468 ±0.02	14.6 ±1.5	0.9357 ±0.04	16.6 ±3.2	0.9168±0.022	29.4 ±1.96	0.868 ±0.016	27.4 ±0.08
DPDT-4	0.947 ±0.027	14.8 ±0.98	0.889 ±0.068	20.6 ±2.94	0.919 ±0.024	24.6 ±2.33	0.853 ±0.016	27.2 ±2.03
QUANT-BNB	0.953 ±0.029	15 ±0	0.817 ±0.023	12.2 ±6.26	0.933 ±0.023	15 ±0	0.859 ±0.011	15 ±0
DT-GFN (ours)	**0.98**±0.04	8.6±3.44	**0.97**±0.03	8.6±1.5	**0.95**±0.02	6.2±0.98	**0.9**±0.002	27±2.83

[2]This would make a sequence model, such as a (decoder-only) transformer (Vaswani et al., 2017; Radford et al., 2018) or a order-invariant version (Lee et al., 2019) a natural choice, but this choice turns out not to be computationally effective, especially at small-to-medium scales.

Baselines and evaluation. We compare our approach against 9 methods belonging to four different families: 1) Bayesian DT sampling algorithms: SMC (Lakshminarayanan et al., 2013), MCMC (Lakshminarayanan et al., 2013), MAPTREE (Sullivan et al., 2024) and BCART (Chipman et al., 1998); 2) methods with explicitly specified splitting criteria: (α^*, β^*)-TSALLIS ENTROPY (Balcan & Sharma, 2024), CART-GINI, CART-ENTROPY; 3) dynamic programming or RL-based methods: DPDT-4 (Hector Kohler, 2025); 4) methods formulating decision tree learning as explicit optimization: QUANT-BNB (Mazumder et al., 2022). We conduct our series of experiments on a variety of widely used tabular datasets from the UCI repository (Dua & Graff, 2019): Iris (Fisher, 1936), Wine (Aeberhard et al., 1992), Breast Cancer Diagnostic (Dua & Graff, 2019) and Raisin (Güvenir & Erel, 2017). For each algorithm, we restrict the chosen maximum tree depth to 5 and we report the average held-out set accuracy and model size over five different train-test splits. Model size is measured by the total number of nodes in a tree. Results are presented in Table 1.

Results. In Table 1, we observe that a decision tree sampled from a trained DT-GFN policy consistently—and often significantly—outperforms state-of-the-art single decision tree construction algorithms from all families. We also show that we obtain trees with the shortest data description length, measured by the average total number of nodes across seeds. Furthermore, when the maximum tree depth is sufficiently large, the minimal complexity comes from the ability of the model to abide by the prior rather than by explicit termination when reaching some small maximum depth.

5.2 BENCHMARKING WITH GREEDY ENSEMBLE METHODS AND DEEP LEARNING METHODS

Table 2: **Test accuracy and model size (total number of tree nodes; where applicable) for ensemble methods, averaged over five random seeds.** For tree methods, including ours, each ensemble contains 1000 trees.

Dataset →	**Iris**		**Wine**		**Breast Cancer**$^{(D)}$		**Raisin**	
Algorithm ↓	TEST ACC↑	SIZE↓	TEST ACC↑	SIZE↓	TEST ACC↑	SIZE↓	TEST ACC↑	SIZE↓
GREEDY RF	0.96±0.01	14.73±0.9	0.9833±0.01	18.89±0.78	0.9474±0.02	33.71±0.93	0.8689±0.01	151.12±2.53
XGBOOST	0.9533±0.03	4.56±0.34	0.9556±0.03	3.51±0.08	0.9579±0.02	6.86±0.16	0.8611±0.02	28.33±0.24
CATBOOST	0.96±0.03	379.771±1.90	0.978±0.01	380.52±0.85	0.9544±0.01	126.72±0.4	0.88±0.02	126.73±0.10
LIGHTGBM	0.9533±0.03	9.39±0.58	**0.9889**±0.01	12.34±0.34	0.9579±0.01	47.04±0.64	0.8622±0.02	59.63±0.17
MLP	0.9667±0.03	N/A	0.8500±0.15	N/A	0.9140±0.01	N/A	0.5167±0.03	N/A
TABTRANSFORMER	0.7933±0.049	N/A	0.978 ±0.021	N/A	0.9316±.0211	N/A	0.8544 ±0.03	N/A
FTTRANSFORMER	0.953±0.016	N/A	0.967±0.02	N/A	**0.961**±0.013	N/A	0.847±0.028	N/A
DT-GFN (ours)	**0.973**±0.04	11.11±2.94	0.983±0.01	8±1.29	0.954±0.02	5.67±0.19	**0.883**±0.03	26.31±1.25

Baselines. We compare our approach to seven methods from two categories: (1) bootstrapped or gradient-boosted trees—GREEDY RF (Breiman, 2001), XGBOOST (Chen & Guestrin, 2016), CATBOOST (Dorogush et al., 2018), and LIGHTGBM (Ke et al., 2017); (2) deep learning models—MLP, TABTRANSFORMER (Huang et al., 2020), and FTTRANSFORMER (Gorishniy et al., 2021). The **experimental setup** and **evaluation criteria** follow §5.1.

Results. In Table 2, we observe that an ensemble constructed from DT-GFN samples with Bayesian model averaging (as per Algorithm 1) is consistently among the most competitive methods while keeping low model complexity.

5.3 EXPERIMENTS IN SYSTEMATIC GENERALIZATION

5.3.1 ROBUSTNESS TO DISTRIBUTION SHIFTS

We consider distribution shifts along two features of the Pima Indians Diabetes dataset (Smith et al., 1988): BMI and Age. The experimental setup is detailed in Appendix H.2. **Baselines** follow §5.2. For tree-based models, results for largest ensembles are reported. All ensembles contain 1000 trees. Ensembles of trees generated by DT-GFN compare favorably to baselines both in-distribution and out-of-distribution under both shifts (Table 3).

Table 3: In-distribution and out-of-distribution test accuracies across two domain shifts.

Domain shift feature →	**BMI**		**Age**	
Algorithm ↓	IN-DIST.	OOD	IN-DIST.	OOD
RF	0.803	0.637	0.8625	0.66
XGBOOST	0.77	0.598	0.788	0.61
CATBOOST	0.803	0.606	0.838	0.645
LIGHTGBM	0.803	0.59	0.825	0.624
MLP	0.7541	0.57	0.825	0.513
TABTRANSFORMER	0.771	0.62	0.825	0.559
FTTRANSFORMER	0.836	0.585	0.775	0.535
DT-GFN (ours)	**0.94**	**0.755**	**0.925**	**0.7**

5.3.2 OUT-OF-DISTRIBUTION DETECTION

Baselines. We compare our approach against seven methods encompassing both recent deep learning algorithms and traditional approaches. Specifically, we include three deep learning methods: (Shenkar & Wolf, 2022), DROCC (Goyal et al., 2020) and GOAD (Bergman & Hoshen, 2020), as well as five

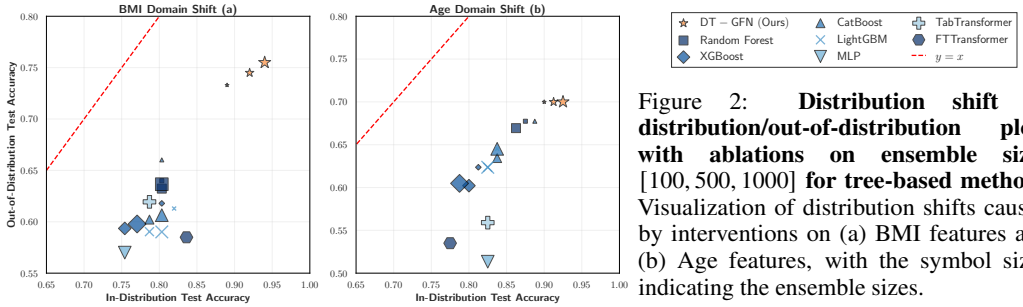

Figure 2: **Distribution shift in-distribution/out-of-distribution plots with ablations on ensemble sizes** $[100, 500, 1000]$ **for tree-based methods.** Visualization of distribution shifts caused by interventions on (a) BMI features and (b) Age features, with the symbol sizes indicating the ensemble sizes.

classical ML methods: COPOD (Li et al., 2020), IFOREST (Liu et al., 2008), KNN (Cover & Hart, 1967), PIDFOREST (Gopalan et al., 2019), and RRCF (Guha et al., 2016). The **experimental setup** and all baseline results are drawn directly from Shenkar & Wolf (2022), as we are able to reproduce their exact results using the provided code.

Evaluation. Shenkar & Wolf (2022) assume *a priori* knowledge of the number of anomalous samples in the test set, say n_a, following a common protocol in anomaly detection (Zong et al., 2018). Then, they adjust the threshold on the prediction metric accordingly so as to select n_a anomalous samples exactly. We outline the details of this procedure in Appendix I. For evaluating our method, we alleviate the need for knowing n_a. For our experiments, as we have access to a trained DT-GFN policy that acts as a as probabilistic model, we can compute the probability of a sample to be normal/anomalous directly. This allows for setting-specific flexibility in designing a classifier, for instance whether we care more about overall accuracy or about being risk-averse, *i.e.*, minimizing misdetections. In our case, we follow a simple procedure where we classify a sample as anomalous if its probability of being normal is at least two standard deviations lower than the average normal class classification probability across samples at test time. DT-GFN results use a single tree. Following Shenkar & Wolf (2022), we use the F1 score as a metric.

Results. In Table 4, we observe that samples of single trees generated by DT-GFN perform (often significantly) better than the state-of-the-art in generalization to detect out-of-distribution samples, while still offering the benefits of a generative probabilistic model and without a priori knowledge of the number of anomalous samples in the test set.

Table 4: **F1 scores on common OOD detection benchmark datasets**, setting is reproduced as per the guidelines in Shenkar & Wolf (2022).

Algorithm ↓ Dataset →	**Thyroid**	**Ecoli**	**Vertebral**	**Glass**
DROCC	0.727	N/A	0.27	0.222
GOAD	0.725	0.693	0.269	0.257
(Shenkar & Wolf, 2022)	0.768	0.7	0.26	0.272
COPOD	0.308	0.256	0.017	0.11
IFOREST	0.789	0.589	0.13	0.11
KNN	0.573	0.778	0.1	0.11
PIDFOREST	0.72	0.256	0.12	0.089
RRCF	0.319	0.289	0.08	0.156
DT-GFN (ours)	**0.794**	**0.812**	**0.404**	**0.315**

5.4 EXPERIMENTS IN CONSISTENT SCALING

We vary ensemble sizes in $\{100, 500, 1000\}$ for tree-based models in experiments in §5.3.1. For each ensemble size, scaling is not only in the samples collected from DT-GFN at inference, instead we train a separate model for each ensemble size configuration. By ensemble size, we do not only mean the samples generated from the DT-GFN policy at inference to construct a predictive ensemble, instead we also describe by that the number of trees DT-GFN generates to compute the TB loss (6). As TB is computed at the level of trajectories, computing it on more trees for the same number of training steps allows it to see more trajectories on average. During inference, we sample the same number of trees used in training. We observe in Fig. 2 that ensembles constructed from trees generated by DT-GFN exhibit properties of systematic and consistent scaling in the ensemble size, *i.e.*, more trees in an ensemble in training results in a increase in generalization both in-distribution and out-of-distribution across two distribution shift instances. On the other hand, gradient-boosted tree algorithms do not scale well with increasing the ensemble size, yielding unstable scaling behavior.

6 DISCUSSION AND FUTURE WORK

DT-GFN is an amortized inference method that generates decision tree models from the Bayesian posterior by sequential construction of decision rules. We have shown that DT-GFN is particularly

strong in settings with few data points, scales well with problem and model size, and is effective in handling distribution shifts. Notable opportunities for future work include extending DT-GFN to online and streaming data scenarios (Chaouki et al., 2024), where GFlowNets have been proposed as a solution (da Silva et al., 2024), and exploring its potential for knowledge-driven Bayesian model selection (Lotfi et al., 2022).

The effectiveness of ensembles of decision trees in systematic generalization, as demonstrated by DT-GFN, makes a case for the use of amortized inference (*e.g.*, using GFlowNets or other deep RL methods) to sample rule-based models: while a neural model is used for parameter inference and Bayesian model averaging, the generated classification models themselves remain lightweight and interpretable. In this sense, DT-GFN is a proof of concept for amortized inference of model structure in more general model classes. It should be built upon in future work on structure inference for probabilistic circuits (of which decision trees are a special case), probabilistic programs, and other structured models.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

Acknowledgments

We thank Tristan Deleu for many valuable discussions in the course of this project. Y.B. acknowledges the support from CIFAR and the CIFAR AI Chair program as well as NSERC funding for the Herzberg Canada Gold medal. This research was enabled in part by computational resources provided by the Digital Research Alliance of Canada (https://alliancecan.ca), Mila (https://mila.quebec), and NVIDIA.

REFERENCES

Stefan Aeberhard, Dionisis Coomans, and Otto de Vel. Wine recognition data. Technical report, UCI Machine Learning Repository, 1992.

Anthropic. The Claude 3 model family: Opus, Sonnet, Haiku, 2024. URL https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Sercan O Arik and Tomas Pfister. TabNet: Attentive interpretable tabular learning. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

Mohammad Azizmalayeri, Ameen Abu-Hanna, and Giovanni Ciná. Unmasking the chameleons: A benchmark for out-of-distribution detection in medical tabular data. *International Journal of Medical Informatics*, 195:105762, 2023.

Maria-Florina Balcan and Dravyansh Sharma. Learning accurate and interpretable decision trees. *Uncertainty in Artificial Intelligence (UAI)*, 2024.

Guilherme Barreto and Ajalmar Neto. Vertebral column database. *UCI Machine Learning Repository*, 2005.

Emmanuel Bengio, Moksh Jain, Maksym Korablyov, Doina Precup, and Yoshua Bengio. Flow network based generative models for non-iterative diverse candidate generation. *Neural Information Processing Systems (NeurIPS)*, 2021.

Yoshua Bengio, Salem Lahlou, Tristan Deleu, Edward J Hu, Mo Tiwari, and Emmanuel Bengio. GFlowNet foundations. *Journal of Machine Learning Research*, 24(210):1–55, 2023.

Liron Bergman and Yedid Hoshen. Classification-based anomaly detection for general data. *International Conference on Learning Representations (ICLR)*, 2020.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Leo Breiman, Jerome Friedman, Charles J Stone, and Richard A Olshen. *Classification and Regression Trees*. Wadsworth International Group, 1984.

Ayman Chaouki, Jesse Read, and Albert Bifet. Online learning of decision trees with Thompson sampling. *Artificial Intelligence and Statistics (AISTATS)*, 2024.

Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. *Knowledge Discovery and Data Mining (KDD)*, 2016.

Hugh A Chipman, Edward I George, and Robert E McCulloch. Bayesian CART model search. *Journal of the American Statistical Association*, 93(443):935–948, 1998.

Hugh A Chipman, Edward I George, and Robert E McCulloch. BART: Bayesian additive regression trees. *Annals of Applied Statistics*, 4(1):266–298, 2010.

Jodie A. Cochrane, Adrian G. Wills, and Sarah J. Johnson. RJHMC-Tree for exploration of the Bayesian decision tree posterior. *arXiv preprint arXiv:2312.01577*, 2023.

Jodie A. Cochrane, Adrian Wills, and Sarah J. Johnson. Divide, conquer, combine Bayesian decision tree sampling. *arXiv preprint arXiv:2403.18147*, 2024.

T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, 1967.

Tiago da Silva, Daniel Augusto de Souza, and Diego Mesquita. Streaming bayes GFlowNets, 2024.

Tristan Deleu, António Góis, Chris Emezue, Mansi Rankawat, Simon Lacoste-Julien, Stefan Bauer, and Yoshua Bengio. Bayesian structure learning with generative flow networks. *Uncertainty in Artificial Intelligence (UAI)*, 2022.

Tristan Deleu, Mizu Nishikawa-Toomey, Jithendaraa Subramanian, Nikolay Malkin, Laurent Charlin, and Yoshua Bengio. Joint Bayesian inference of graphical structure and parameters with a single generative flow network. *Neural Information Processing Systems (NeurIPS)*, 2023.

Tristan Deleu, Padideh Nouri, Nikolay Malkin, Doina Precup, and Yoshua Bengio. Discrete probabilistic inference as control in multi-path environments. *Uncertainty in Artificial Intelligence (UAI)*, 2024.

Emir Demirović, Anna Lukina, Emmanuel Hebrard, Jeffrey Chan, James Bailey, Christopher Leckie, Kotagiri Ramamohanarao, and Peter J Stuckey. Murtree: Optimal decision trees via dynamic programming and search. *Journal of Machine Learning Research*, 23(26):1–47, 2022.

Matthew F. Dixon, Igor Halperin, and Paul Bilokon. *Machine Learning in Finance*. Springer, 2020.

Anna Veronika Dorogush, Vasily Ershov, and Andrey Gulin. CatBoost: Gradient boosting with categorical features support. *Neural Information Processing Systems (NIPS)*, 2018.

Dheeru Dua and Casey Graff. Uci machine learning repository, 2019.

Ian W Evett and Ernest J Spiehler. Glass identification database. *UCI Machine Learning Repository*, 1987.

R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of eugenics*, 7(2): 179–188, 1936.

Josh Gardner, Zoran Popovic, and Ludwig Schmidt. Benchmarking distribution shift in tabular data with TableShift. *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

Abhinav Garlapati, Aditi Raghunathan, Vaishnavh Nagarajan, and Balaraman Ravindran. A reinforcement learning approach to online learning of decision trees. *arXiv preprint arXiv:1507.06923*, 2015.

Parikshit Gopalan, Vatsal Sharan, and Udi Wieder. PIDForest: Anomaly detection via partial identification. *Neural Information Processing Systems (NeurIPS)*, 2019.

Yury Gorishniy, Ivan Rubachev, Valentin Khrulkov, and Artem Babenko. Revisiting deep learning models for tabular data. *Neural Information Processing Systems (NeurIPS)*, 2021.

Sachin Goyal, Aditi Raghunathan, Moksh Jain, Harsha Vardhan Simhadri, and Prateek Jain. DROCC: Deep robust one-class classification. *International Conference on Machine Learning (ICML)*, 2020.

Sarah Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on tabular data? *Neural Information Processing Systems (NeurIPS)*, 2022.

Sudipto Guha, Nina Mishra, Gourav Roy, and Okke Schrijvers. Robust random cut forest based anomaly detection on streams. *International Conference on Machine Learning (ICML)*, 2016.

Agrim Gupta, Ali Razavi, Andeep Toor, Ankush Gupta, Dumitru Erhan, Eleni Shaw, Eric Lau, Frank Belletti, Gabe Barth-Maron, Gregory Shaw, Hakan Erdogan, Hakim Sidahmed, Henna Nandwani, Hernan Moraldo, Hyunjik Kim, Irina Blok, Jeff Donahue, José Lezama, Kory Mathewson, Kurtis David, Matthieu Kim Lorrain, Marc van Zee, Medhini Narasimhan, Miaosen Wang, Mohammad Babaeizadeh, Nelly Papalampidi, Nick Pezzotti, Nilpa Jha, Parker Barnes, Pieter-Jan Kindermans, Rachel Hornung, Ruben Villegas, Ryan Poplin, Salah Zaiem, Sander Dieleman, Sayna Ebrahimi, Scott Wisdom, Serena Zhang, Shlomi Fruchter, Signe Nørly, Weizhe Hua, Xinchen Yan, Yuqing Du, and Yutian Chen. Veo 2. 2024. URL https://deepmind.google/technologies/veo/veo-2/.

H.A. Güvenir and E. Erel. Raisin dataset. *UCI Machine Learning Repository*, 2017.

David J. Hand. Classifier Technology and the Illusion of Progress. *Statistical Science*, 21(1):1–14, 2006.

Philippe Preux Hector Kohler, Riad Akrour. Breiman meets bellman: Non-greedy decision trees with mdps. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2025.

David Holzmüller, Léo Grinsztajn, and Ingo Steinwart. Better by default: Strong pre-tuned MLPs and boosted trees on tabular data. *Neural Information Processing Systems (NeurIPS)*, 2024.

Paul Horton and Kenta Nakai. A probabilistic classification system for predicting the cellular localization sites of proteins. *Intelligent Systems in Molecular Biology*, pp. 109–115, 1996.

Edward J Hu, Nikolay Malkin, Moksh Jain, Katie Everett, Alexandros Graikos, and Yoshua Bengio. GFlowNet-EM for learning compositional latent variable models. *International Conference on Machine Learning (ICML)*, 2023.

Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Neural Information Processing Systems (NeurIPS)*, 2019.

Xin Huang, Ashish Khetan, Milan Cvitkovic, and Zohar Karnin. TabTransformer: Tabular data modeling using contextual embeddings. *arXiv preprint arXiv:2012.06678*, 2020.

Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. LightGBM: A highly efficient gradient boosting decision tree. *Neural Information Processing Systems (NIPS)*, 2017.

Guolin Ke, Di Wang, and Haifeng Zheng. DeepGBM: A deep learning framework distilled by GBDT for online prediction tasks. *Knowledge Discovery and Data Mining (KDD)*, 2019.

Güünter Klambauer, Thomas Unterthiner, Andreas Mayr Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Neural Information Processing Systems (NIPS)*, 2017.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, Tony Lee, Etienne David, Ian Stavness, Wei Guo, Berton A. Earnshaw, Imran S. Haque, Sara Beery, Jure Leskovec, Anshul Kundaje, Emma Pierson, Sergey Levine, Chelsea Finn, and Percy Liang. WILDS: A benchmark of in-the-wild distribution shifts. *International Conference on Machine Learning (ICML)*, 2021.

Balaji Lakshminarayanan, Daniel M. Roy, and Yee Whye Teh. Top-down particle filtering for bayesian decision trees. *International Conference on Machine Learning (ICML)*, 2013.

Balaji Lakshminarayanan, Daniel M Roy, and Yee Whye Teh. Mondrian forests: Efficient online random forests. *Neural Information Processing Systems (NIPS)*, 2016.

Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiorek, Seungjin Choi, and Yee Whye Teh. Set transformer: A framework for attention-based permutation-invariant neural networks. *International Conference on Machine Learning (ICML)*, 2019.

Wenqian Li, Yinchuan Li, Zhigang Li, Jianye Hao, and Yan Pang. DAG Matters! GFlowNets enhanced explainer for graph neural networks. *International Conference on Learning Representations (ICLR)*, 2023.

Zheng Li, Yue Zhao, Nicola Botta, Cezar Ionescu, and Xiyang Hu. COPOD: copula-based outlier detection. *International Conference on Data Mining*, 2020.

Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. *International Conference on Machine Learning (ICML)*, 2020.

Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. Isolation forest. *International Conference on Data Mining*, 2008.

Sanae Lotfi, Pavel Izmailov, Gregory Benton, Micah Goldblum, and Andrew Gordon Wilson. Bayesian model selection, the marginal likelihood, and generalization. *International Conference on Machine Learning (ICML)*, 2022.

David JC MacKay. *Information theory, inference and learning algorithms*. Cambridge university press, 2003.

Nikolay Malkin, Moksh Jain, Emmanuel Bengio, Chen Sun, and Yoshua Bengio. Trajectory balance: Improved credit assignment in GFlowNets. *Neural Information Processing Systems (NeurIPS)*, 2022.

Nikolay Malkin, Salem Lahlou, Tristan Deleu, Xu Ji, Edward Hu, Katie Everett, Dinghuai Zhang, and Yoshua Bengio. GFlowNets and variational inference. *International Conference on Learning Representations (ICLR)*, 2023.

Rahul Mazumder, Xiang Meng, and Haoyue Wang. Quant-BnB: A scalable branch-and-bound method for optimal decision trees with continuous features. *International Conference on Machine Learning (ICML)*, 2022.

Duncan McElfresh, Sujay Khandagale, Jonathan Valverde, Vishak Prasad C, Benjamin Feuer, Chinmay Hegde, Ganesh Ramakrishnan, Micah Goldblum, and Colin White. When do neural nets outperform boosted trees on tabular data? *Neural Information Processing Systems (NeurIPS) Datasets and Benchmarks Track*, 2023.

John Mingers. Expert systems—rule induction with statistical data. *Journal of the operational research society*, 38(1):39–47, 1987.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2024.

Sergei Popov, Stanislav Morozov, and Artem Babenko. Neural oblivious decision ensembles for deep learning on tabular data. *International Conference on Learning Representations (ICLR)*, 2020.

Karol Przystalski and Rohit M. Thanki. *Medical Tabular Data*. Springer International Publishing, 2024.

J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

J. Ross Quinlan. Simplifying decision trees. *International journal of man-machine studies*, 27(3): 221–234, 1987.

J Ross Quinlan. *C4. 5: programs for machine learning*. Elsevier, 2014.

J Ross Quinlan et al. Thyroid disease database. *UCI Machine Learning Repository*, 1987.

Joaquin Quiñonero-Candela, Masashi Sugiyama, Anton Schwaighofer, and Neil D Lawrence. *Dataset shift in machine learning*. Mit Press, 2022.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI Blog*, 1(8), 2018.

Shebuti Rayana. Odds library, 2016. URL https://odds.cs.stonybrook.edu.

Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

Tom Shenkar and Lior Wolf. Anomaly detection for tabular data with internal contrastive learning. *International Conference on Learning Representations (ICLR)*, 2022.

Ravid Shwartz-Ziv and Amitai Armon. Tabular data: Deep learning is not all you need. *Information Fusion*, 81:84–90, 2022.

J.W. Smith, J.E. Everhart, W.C. Dickson, W.C. Knowler, and R.S. Johannes. Pima indians diabetes database. Technical report, National Institute of Diabetes and Digestive and Kidney Diseases, 1988.

Colin Sullivan, Mo Tiwari, and Sebastian Thrun. MAPTree: Beating "Optimal" decision trees with Bayesian decision trees. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2024.

Daniil Tiapkin, Nikita Morozov, Alexey Naumov, and Dmitry P Vetrov. Generative flow networks as entropy-regularized RL. *Artificial Intelligence and Statistics (AISTATS)*, 2024.

Nicholay Topin, Stephanie Milani, Fei Fang, and M. Veloso. Iterative bounding MDPs: Learning interpretable policies via non-interpretable methods. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Neural Information Processing Systems (NIPS)*, 2017.

W.H. Wolberg, W.N. Street, and O.L. Mangasarian. Breast cancer cytology dataset. *UCI Machine Learning Repository*, 1995.

Yuhong Wu, Håkon Tjelmeland, and Mike West. Bayesian CART: Prior specification and posterior simulation. *Journal of Computational and Graphical Statistics*, 16(1):44–66, 2007.

Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. VIME: Extending the success of self- and semi-supervised learning to tabular domain. *Neural Information Processing Systems (NeurIPS)*, 2020.

Taraneh Younesian, Daniel Daza, Emile van Krieken, Thiviyan Thanapalasingam, and Peter Bloem. GRAPES: Learning to sample graphs for scalable graph neural networks. *arXiv preprint arXiv:2310.03399*, 2024.

Dinghuai Zhang, Hanjun Dai, Nikolay Malkin, Aaron Courville, Yoshua Bengio, and Ling Pan. Let the flows tell: Solving graph combinatorial problems with GFlowNets. *Neural Infromation Processing Systems (NeurIPS)*, 2023.

Mingyang Zhou, Zichao Yan, Elliot Layne, Nikolay Malkin, Dinghuai Zhang, Moksh Jain, Mathieu Blanchette, and Yoshua Bengio. PhyloGFN: Phylogenetic inference with generative flow networks. *International Conference on Learning Representations (ICLR)*, 2024.

Heiko Zimmermann, Fredrik Lindsten, Jan-Willem van de Meent, and Christian A. Naesseth. A variational perspective on generative flow networks. *Transactions on Machine Learning Research (TMLR)*, 2023.

Bo Zong, Qi Song, Martin Renqiang Min, Wei Cheng, Cristian Lumezanu, Daeki Cho, and Haifeng Chen. Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. *International Conference on Learning Representations (ICLR)*, 2018.

A Illustrating Example- Pitfalls of Greedy Decision Trees and Ensemble Methods

To motivate part of our work, we devise a simple setting that showcases some of the pitfalls of greedy tabular methods, such as greedily constructed decision trees, and subsequent ensemble methods. We start by creating a variant of what is typically referred to as the hidden XOR problem, where we construct a synthetic dataset which has 20 features, 2 of which are binary and the rest are randomly generated real features. The label for each data point is simply an XOR operation between the two binary features, while the rest of the features are completely irrelevant for the classification task. Typically, a sufficiently expressive training dataset in this setting would have a minimum of 4 data points, enumerating all possible relevant input-output pairs, assuming we are not trying to test if a given model can generalize beyond the seen training distribution modes. In essence, we would like to examine a behavior trends that allow us to control the task difficulty, while observing how methods scale in that. In particular, we show how simply scaling the number of irrelevant features affects performance, with a dataset size of $|\mathcal{D}| = 1000$.

Scaling the number of noise features. Varying the number of irrelevant or *noise* features allows to progressively control the task difficulty, and examine how different methods behave under varying amounts of noise, *i.e.*, a larger model search space, but no underlying structure being added to the data generating process. On the left of Fig. 3, the chosen noise features are binary, which significantly restricts the search space over decision trees. For instance, even when consider a total of 20 binary features, there are only 8,000 possible distinct decision trees of depth 2 that can be constructed given the data (a decision tree of depth 2 is enough to reconstruct the label generating process). We observe perfect test accuracy on all models for a small number of noise features. While a GFlowNets maintain perfect test accuracy as the number of noise features gets larger, we see a slight drop in test accuracy for greedy RFs for 20 features and even more significant drop for SMC as the number of features increases. Note that for 10 features, the number of possible distinct decision trees of depth 2 is only 1000 (as shown in §3.1). On the right of Fig. 3, the noise features are now randomly generated real numbers, which significantly enlarges the search space over decision tree models. We observe that test accuracy drops significantly for both greedy RFs and SMC when increasing the number of noise features, while GFlowNets maintain perfect test accuracy. We argue here, through presenting a simple task where the label generating process does not change but only the number of noise features increases, that structure inference is crucial to learning and mitigating many undesirable characteristics of greedy methods.

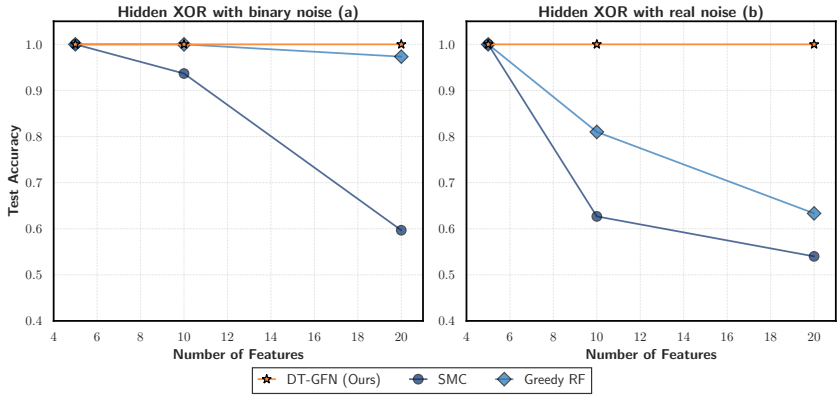

Figure 3: **Varying the number of features in a hidden XOR task where the label is an XOR operation between two features.** Noise features are chosen to be either binary (**left**) or real (**right**). All datasets contain 1000 samples.

B Related Work

Learning decision trees with explicit *splitting* criteria. Decision trees have historically been constructed top-down via splitting the feature space sequentially according to some heuristic criterion (Breiman et al., 1984; Quinlan, 1986; 2014), usually based on some form of entropy reduction coupled with principled pruning (Quinlan, 1987; Mingers, 1987). Recent work (Balcan & Sharma, 2024) splits based on a generalization to Tsallis entopy with a regularizer term on the number of tree leaves, arguing that less complex models tend to be more interpretable and generalize better. Our work learns a *trained policy* that makes splitting decisions entirely informed by data, up to priors.

Tabular deep learning. Diverse architectures and learning representations have been proposed to model tabular data (*e.g.*, Klambauer et al. (2017); Popov et al. (2020); Ke et al. (2019); Yoon et al. (2020); Gorishniy et al. (2021); Arik & Pfister (2021); Shwartz-Ziv & Armon (2022)). Our work uses deep learning to parametrize a decision-making policy over decision rules, yielding a generative model over decision trees, instead of modeling data directly.

Learning decision trees as Bayesian inference. Fundamental approaches formulate decision tree learning as posterior inference (Chipman et al., 1998; 2010; Lakshminarayanan et al., 2013; 2016), where the latter is determined by the likelihood of data under the tree (given some tree structure) with some prior on possible tree structures. The main challenge in such task is the intractability of the search space over decision trees, even in relatively toy settings as shown in §3.1. Our work amortizes this search and leverages exploration strategies from off-policy deep RL, more specifically GFlowNets, to efficiently sample from different modes of the posterior, yielding a diverse set of high-quality tree structures that capture distinct decision-making patterns in data.

Learning decision trees as dynamic programming and/or explicit optimization. Recent work framed decision tree learning as a structured optimization problem, with Hu et al. (2019) formulating it as a Markov decision process (MDP) via mixed-integer programming. Lin et al. (2020) extended this to both classification and regression with optimality guarantees, and Demirović et al. (2022) introduced efficient dynamic programming decompositions for multi-task objectives. Garlapati et al. (2015) formulated the learning of decision trees with ordinal attributes as an MDP, and Topin et al. (2021) introduced an iterative bounding MDP to enable deep reinforcement learning algorithms to learn decision-tree policies. Mazumder et al. (2022) further leveraged branch-and-bound to handle continuous attributes more effectively. Our work considers a different optimization problem, which is fitting the Bayesian posterior over decision trees.

Amortized inference and GFlowNets. *Generative Flow Networks* (GFlowNets; Bengio et al. (2021; 2023)) are a family of amortized variational inference algorithms (Malkin et al., 2023; Zimmermann et al., 2023) that formulate the problem of sampling from a target unnormalized density as a sequential decision-making process and solve it by methods related to entropy-regularized reinforcement learning (Tiapkin et al., 2024; Deleu et al., 2024). GFlowNets have been used as amortized posterior samplers in numerous applications, including those over graphs and similar structures, *e.g.*, causal models (Deleu et al., 2022; 2023), parse trees (Hu et al., 2023), phylogenetic trees (Zhou et al., 2024), and subgraph structures (Li et al., 2023; Zhang et al., 2023; Younesian et al., 2024). Our work uses GFlowNets as the main (deep RL) training algorithm to construct our decision-making policy.

C Mathematical Glossary

We aim to make the manuscript rather self-contained, hence we introduce a few mathematical concepts that would provide a clear insight into some of the methods we consider throughout the paper, either as our own or ones that we benchmark with.

Γ-function. For a real number $z > 0$, the Γ function is defined as an improper integral as follows

$$\Gamma(z) = \int_0^\infty t^{z-1} e^{-t}\, dt$$

Tsallis Entropy. Let $P = \{p_i\}$ be a discrete probability distribution, where $p_i \geq 0$ and $\sum_i p_i = 1$. The Tsallis entropy of P with entropic index $q \in \mathbb{R}$ is defined as

$$S_q(P) = \frac{1}{q-1}\left(1 - \sum_i p_i^q\right), \quad q \neq 1.$$

In the limit as $q \to 1$, Tsallis entropy reduces to the classical Shannon entropy

$$S_1(P) = -\sum_i p_i \log p_i.$$

D Proofs and Auxiliary Results

We outline proofs of our theoretical claims, along with reasoning for some of the design choices that we use for our construction.

D.1 Proofs

We reiterate that the results in this section were already proved in (Chipman et al., 1998) in some form, we reproduce them given our setting and notation for the convenience of the reader.

Proposition D.1 (Likelihood of a Bayesian DT)**.** *The likelihood under a Bayesian decision tree (T, Θ) given features $\mathcal{X}$ and labels $\mathcal{Y}$ is written as follows*

$$\mathbb{P}\left[\mathcal{Y}|\mathcal{X}, T, \Theta\right] = \prod_{\ell \in \mathcal{L}} \prod_{i \in \Delta(\ell)} \prod_{c \in C} \theta_{\ell,(c)}^{\mathbb{I}^{(c)}(y_i)} = \prod_{\ell \in \mathcal{L}} \prod_{c \in C} \theta_{\ell,c}^{\sum_{i \in \Delta(\ell)} \mathbb{I}^{(c)}(y_i)},$$

where $\mathcal{L}$ is the set of leaves in the tree T, C is the set of classes, $\theta_{\ell,(c)}$ is the probability of sampling class c under leaf ℓ, and $\mathbb{I}^{(c)}(y_i)$ is the indicator function of whether y_i belongs to class c.

Proof of Prop. 3.1. By construction, a decision tree T partitions data into $|\mathcal{L}|$ splits, *i.e.*, $\Delta(\ell)$ for $\ell \in \mathcal{L}$ where $\mathcal{L}$ is the set of leaves under the decision tree. Each leaf ℓ under the tree induces an independent probability distribution Dirichlet($\boldsymbol{\theta}_\ell$) over the probability of a given label y_i occurring under leaf ℓ. Along this line of reasoning, the probability of picking a given label y_i is conditionally independent both of $\mathcal{X}$ and of the other classes in $\mathcal{Y}$ given the leaf ℓ it lands on (which is only determined from T and $\mathbf{x}_i$) and the corresponding classification parameter $\boldsymbol{\theta}_\ell$ (which, again, is determined from Θ and ℓ only). The latter observation justifies all the steps in our proof, which we formulate in the following

$$\begin{aligned}
\mathbb{P}[\mathcal{Y}|\mathcal{X}, T, \Theta] &= \prod_{i \in |\mathcal{Y}|} \mathbb{P}[y_i|\mathbf{x}_i, T, \Theta] \\
&= \prod_{\ell \in \mathcal{L}} \prod_{i \in \Delta(\ell)} \mathbb{P}[y_i|\boldsymbol{\theta}_\ell] \\
&= \boxed{\prod_{\ell \in \mathcal{L}} \prod_{c \in C} \theta_{\ell,c}^{\sum_{i \in \Delta(\ell)} \mathbb{I}^{(c)}(y_i)}},
\end{aligned}$$

which concludes the proof. □

Proposition D.2 (Marginal Likelihood of a Bayesian DT)**.** *Assuming $\boldsymbol{\theta} \sim$ Dirichlet($\boldsymbol{\alpha}$), the marginal likelihood of a Bayesian decision tree T given features $\mathcal{X}$ and labels $\mathcal{Y}$*

$$\mathbb{P}\left[\mathcal{Y}|\mathcal{X},T\right] = \left(\frac{\Gamma(\sum_{c\in\mathcal{C}}\alpha_c)}{\prod_{c\in\mathcal{C}}\Gamma(\alpha_c)}\right)^{|\mathcal{L}|}\prod_{\ell\in\mathcal{L}}\frac{\prod_{c\in\mathcal{C}}\Gamma(n_{\ell,c}+\alpha_c)}{\Gamma(n_\ell+\sum_{c\in\mathcal{C}}\alpha_c)},$$

where $n_{\ell,c} = \sum_{i\in\Delta(\ell)}\mathbb{I}^{(c)}(y_i)$ *is the empirical count of data points with label* c *at leaf* ℓ *and* $n_\ell = \sum_{c\in\mathcal{C}} n_{\ell,c}$ *is the total empirical count of data points (of all classes) at leaf* ℓ.

Proof of Prop. 3.2. We would like to derive $\mathbb{P}[\mathcal{Y}|\mathcal{X},T]$ by marginalizing over Θ. Assuming $\boldsymbol{\theta} \sim \text{Dirichlet}(\boldsymbol{\alpha})$, we prove the result of that, which we present in Prop. 3.2, in the following

$$\begin{aligned}
\mathbb{P}[\mathcal{Y}|\mathcal{X},T] &= \int_\Theta \mathbb{P}[\mathcal{Y}|\mathcal{X},T,\Theta]\cdot p[\Theta]\, d\Theta \\
&= \int_\Theta \prod_{\ell\in\mathcal{L}}\prod_{c\in\mathcal{C}} \theta_{\ell,c}^{\sum_{i\in\Delta(\ell)}\mathbb{I}^{(c)}(y_i)}\cdot\left(\frac{\theta_{\ell,c}^{\alpha_c}}{\frac{\prod_{c\in\mathcal{C}}\Gamma(\alpha_c)}{\Gamma(\sum_{c\in\mathcal{C}}\alpha_c)}}\right) d\Theta \\
&= \left(\frac{\Gamma(\sum_{c\in\mathcal{C}}\alpha_c)}{\prod_{c\in\mathcal{C}}\Gamma(\alpha_c)}\right)^{|\mathcal{L}|}\int_\Theta \prod_{\ell\in\mathcal{L}}\prod_{c\in\mathcal{C}} \theta_{\ell,c}^{\alpha_c+\sum_{i\in\Delta(\ell)}\mathbb{I}^{(c)}(y_i)}\, d\Theta \\
&= \left(\frac{\Gamma(\sum_{c\in\mathcal{C}}\alpha_c)}{\prod_{c\in\mathcal{C}}\Gamma(\alpha_c)}\right)^{|\mathcal{L}|}\prod_{\ell\in\mathcal{L}}\int_{\boldsymbol{\theta}_\ell}\prod_{c\in\mathcal{C}} \theta_{\ell,c}^{\alpha_c+\sum_{i\in\Delta(\ell)}\mathbb{I}^{(c)}(y_i)}\, d\boldsymbol{\theta}_\ell \\
&= \boxed{\left(\frac{\Gamma(\sum_{c\in\mathcal{C}}\alpha_c)}{\prod_{c\in\mathcal{C}}\Gamma(\alpha_c)}\right)^{|\mathcal{L}|}\prod_{\ell\in\mathcal{L}}\frac{\prod_{c\in\mathcal{C}}\Gamma(n_{\ell,c}+\alpha_c)}{\Gamma(n_\ell+\sum_{c\in\mathcal{C}}\alpha_c)}}
\end{aligned}$$

where $n_{\ell,c} = \sum_{i\in\Delta(\ell)}\mathbb{I}^{(c)}(y_i)$ is the empirical count of data points with label c at leaf ℓ and $n_\ell = \sum_{c\in\mathcal{C}} n_{\ell,c}$ is the total empirical count of data points (of all classes) at leaf ℓ, which concludes our proof. □

D.2 Prior over Decision Tree Structure and Choice of β

To choose an appropriate prior β, we need to ensure that β properly normalizes the likelihood $\mathbb{P}[\mathcal{Y}|\mathcal{X},T]$, *i.e.*, accurately approximates the number of tree structures T that would model $\mathcal{X}$. Alternatively, from a coding theory perspective, β could also be interpreted as the average code length of a tree structure modeling $\mathcal{X}$. The number of possible binary tree structures with n internal nodes is given by the n-th Catalan number, C_n, which asymptotically behaves as

$$C_n \sim \frac{4^n}{n^{3/2}}.$$

The corresponding coding length for that is

$$\log(C_n) \approx n\log(4) - \frac{3}{2}\log(n).$$

Now, for each of the n nodes, we have to pick a splitting feature out of d variables and one of t possible thresholds, the complexity of such a choice on average is

$$n(\log(p)+\log(t)).$$

Overall, the total coding length $\ell(n)$ for a tree with n decision nodes is

$$\ell(n) \approx n\Big(\log(4)+\log(p)+\log(t)\Big) - \frac{3}{2}\log(n).$$

Hence a prior favoring trees with shorter description lengths can be expressed as

$$\begin{aligned}
\mathbb{P}[T|\mathcal{X}] &\propto \exp\Big(-\ell(n)\Big) \\
&\propto \exp\Big(-n\Big[\log(4)+\log(p)+\log(t)\Big] + \frac{3}{2}\log(n)\Big)
\end{aligned}$$

Considering the asymptotically dominant term, we should choose β such that

$$\boxed{\beta \sim \log(4)+\log(p)+\log(t)}$$

D.3 Reward Computation in Mini-Batches

Computing the reward for a given tree, as per §4.2, requires recursing over the tree $|\mathcal{X}|$ times in the worst case. Given how large $|\mathcal{X}|$ in our considered setting can get, an important result to scaling our approach is the ability to compute rewards in mini-batches. As shown already in (Bengio et al., 2023), computing the rewards in mini-batches ensures that the optimal policy $\hat{\mathbb{P}}_F$ minimizing the TB square-loss for all complete trajectories converges to the true reward/posterior in expectation.

E Construction of Ensembles of Predictors

Given access to a trained DT-GFN policy, we perform a prediction using samples from that as follows.

Algorithm 1 Bayesian Ensemble Prediction with DT-GFN Samples

Input. Data point $\mathbf{x}_j$, set of decision tree samples $\{T_i\}$, dataset $\mathcal{D}$
Output. Predicted class $\hat{y}_j$
for each tree T_i **do**
  Compute $\mathbb{P}(y_j = c \mid \mathbf{x}_j, T_i)$ for all classes c using the GFlowNet policy.
end for
Compute $\log\left[\mathbb{P}(T_i|\mathcal{D})\right]$ for all trees T_i
$m \leftarrow \max_i \log\left[\mathbb{P}(T_i|\mathcal{D})\right]$
$\log\left[\mathbb{P}(\mathcal{D})\right] \leftarrow \log\left(\sum_k \exp(\log\left[\mathbb{P}(T_k|\mathcal{D})\right] - m)\right) + m$
for each tree T_i **do**
  $\mathbb{P}(T_i|\mathcal{D}) \leftarrow \frac{\exp(\log[\mathbb{P}(T_i|\mathcal{D})])}{\sum_k \exp(\log[\mathbb{P}(T_k|\mathcal{D})])}$
end for
for each class c **do**
  $\mathbb{P}(y_j = c|\mathbf{x}_j, \mathcal{D}) \leftarrow \sum_i \mathbb{P}(T_i|\mathcal{D}) \cdot \mathbb{P}(y_j = c|\mathbf{x}_j, T_i)$
end for
$\hat{y}_j \leftarrow \arg\max_c \mathbb{P}(y_j = c|\mathbf{x}_j, \mathcal{D})$
return $\hat{y}_j$

F TRAINING DETAILS

We list a variety of strategies for GFlowNet training we use throughout the paper along with setup-specific hyperparameters allowing to reproduce our results.

F.1 EXPLORATION STRATEGY

To allow for exploration throughout training, we employ the following strategies, which we find to consistently perform well across our considered datasets and tasks.

ϵ-greedy annealing. We generate a set of trajectories from the DT-GFN policy throughout training, such that actions are sampled with probability $1 - \epsilon$ according to the policy and with probability ϵ uniformly at random. ϵ is varied throughout training from some ϵ_0 predefined at initialization to some small positive constant, *i.e.*, $\epsilon \in (0, \epsilon_0]$. ϵ_0 is set to 0.1 in our experiments.

Replay buffer. We use a replay buffer to store the Top-K "best" trees (with highest rewards) we have seen so far in training. Then, we sample trajectories from the latter using the backward policy P_B, simply set to a uniform distribution over parent states at each step starting from a given terminal state.

F.2 HYPERPARAMETERS

We list hyperparameters we consistently use throughout our experiments in Table 5. We highlight that it is also possible to use smaller stopping depths for faster training, or a smaller discretization threshold constant for datasets lower precisions, for instance only 1 for binary hidden XOR or 9 for Iris.

Table 5: **Training hyperparameters for reproducing our experiments.**

Hyperparameter	**Value**
Tree Construction	
Max Tree Depth	5
Thresholds Discretization	99
Number of Samples	1000
Policy	
Policy Model	MLP
Hidden Layers	3
Hidden Units per Layer	256
Optimization	
Learning Rate	0.01
Training Steps	100
Batch Size (Forward)	90
Batch Size (Backward Replay)	10
Exploration	
Replay Buffer Capacity	100
Random Action Probability	0.1
Proxy	
Parameter Prior ($\boldsymbol{\alpha}$)	$\boldsymbol{\alpha} = [0.1]\times$ (number of classes)
Structure Prior (β)	$\log(4) + \log(d)$

G Empirical Cost Analysis

G.1 Budget-constrained Training

We would like to test how DT-GFN would perform under a tight time/compute budget. As inference costs are negligible compared to training costs (see Table 7), it would be interesting to observe: (i) how the accuracy of our predictions vary with training time budgets of $[0, 10, 20]$ (seconds) (ii) how ensemble predictions with Algorithm 1 vary in ensemble size given varying levels of structure inference training, from no time (random structure from a base DT-GFN policy) to a DT-GFN policy trained for 20 seconds. Experiment are carried on the Iris dataset, in the same setup as §5.2; results are averaged over data split seeds $[1, 2, 3, 4, 5]$.

We observe that DT-GFN manages to get high test accuracies, with **consistent scaling across both training time and ensemble size**. Notably, even with random structure, ensembles constructed according to Algorithm 1 still manage to scale consistently. For the maximum training budget of 20 seconds, all DT-GFN ensembles outperform the best gradient-boosted tree baseline (GBT) and match the best baseline in §5.2. We further highlight that at a budget of 50 seconds, test accuracy plateaus at the current maximum for all ensemble sizes. Yet, these results are still lower than DT-GFN's results in §5.2 with more resources (see Table 5), further hinting at scaling capacity with increase in resources.

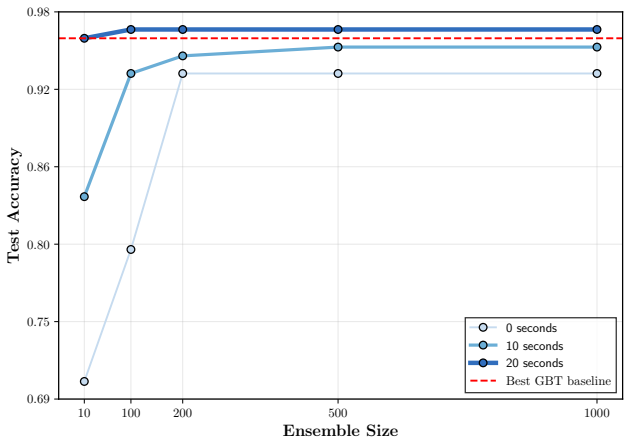

Figure 4: **DT-GFN scaling with ensemble size in** $[10, 100, 200, 500, 1000]$ **and allocated time/compute budget in [0 seconds, 10 seconds, 20 seconds]**. Experiment performed on the Iris dataset and results are averaged over data split seeds $[1, 2, 3, 4, 5]$. We show **consistent** scaling across both ensemble sizes and training time/compute resources. Inference costs are negligible as shown in Table 7.

Reproducibility. Hyperparameters to reproduce our results are similar to Table 5, with exceptions highlighted in Table 6.

Table 6: **Training hyperparameters for reproducing our experiments.** `${variable}` denotes the varying scaling axes of Fig. 4

Hyperparameter	Value
Tree Construction	
Max Tree Depth	3
Thresholds Discretization	9
Number of Samples	`${variable}`
Optimization	
Training Steps	`${variable}`
Exploration	
Replay Buffer Capacity	10
Random Action Probability	0.05

G.2 Finite-time Amortization Cost Measurements

We highlight how training and inference costs scale in the most "costly" training parameters we observe, which are tree max. depths and the number of trees (proportionally trajectories) the DT-GFN policy is trained on.

Table 7: **Training cost per epoch and inference cost (in seconds) for sampling an ensemble of DTs of varying size at different maximum depths.**

		Ensemble Size			
		1		100	
Dataset	**Max. Depth**	**Train**	**Infer.**	**Train**	**Infer.**
Iris	2	1.09	0.0074	1.09	0.4239
	4	4.21	0.0169	4.29	0.4468
Wine	2	1.36	0.0032	1.85	0.283
	4	3.95	0.0047	4.50	0.4953
Breast Cancer$^{(D)}$	2	1.57	0.0073	2.26	0.195
	4	3.31	0.0145	3.73	0.5303
Raisin	2	1.79	0.015	2.47	0.23
	4	5.05	0.0037	6.23	0.4804

G.3 Hardware

Our training is conducted on an RTX 8000 GPU with 16GB of allocated memory. The training cost is measured on this hardware setup. Notably, the training process is lightweight and can be easily fitted on a local machine with minimal computational resources.

H Further Details on Experimental Setup

H.1 Datasets

Each dataset contains both numerical and categorical features, and the task in all datasets is classification. For the feature support to be tractable for discretization, we scale all features to $[0, 1]$ with Min-Max scaling and discretization is performed uniformly. We list the datasets and splits we use in Table 9.

Table 8: **Dataset characteristics and reproducibility.** For systematic generalization experiments, test columns in the form a/b denote number of (a) in-distribution and (b) out-of-distribution samples respectively.

Dataset	**n**	**d**	**Train**	**Test**	**Split Seeds**	**Experiment**
Iris (Fisher, 1936)	150	4	120	30	[1, 2, 3, 4, 5]	DT & Ensemble
Wine (Aeberhard et al., 1992)	178	13	142	36	[1, 2, 3, 4, 5]	DT & Ensemble
Breast Cancer$^{(D)}$ (Wolberg et al., 1995)	569	30	455	114	[1, 2, 3, 4, 5]	DT & Ensemble
Raisin (Güvenir & Erel, 2017)	900	7	720	180	[1, 2, 3, 4, 5]	DT & Ensemble
Pima (Smith et al., 1988) **(BMI)**	768	8	243	60/465	[42]	Domain shift
Pima (Smith et al., 1988) **(Age)**	768	8	316	80/372	[42]	Domain shift
Thyroid (Quinlan et al., 1987)	3772	6	1840	1839/93	Shenkar & Wolf (2022)	OOD detection
Ecoli (Horton & Nakai, 1996)	336	7	164	163/9	Shenkar & Wolf (2022)	OOD detection
Vertebral (Barreto & Neto, 2005)	240	6	106	104/30	Shenkar & Wolf (2022)	OOD detection
Glass (Evett & Spiehler, 1987)	214	9	103	102/9	Shenkar & Wolf (2022)	OOD detection

H.2 Preprocessing

For the first two sets of datasets, we use MinMax scaling for scaling features back to $[0, 1]$. All categorical features are encoded using OneHotEncoding. For the third set of datasets, which we use for OOD detection, we rely on the preprocessing protocol of (Shenkar & Wolf, 2022). All datasets are obtained from the UCI repository (Dua & Graff, 2019) except the data for ODD detection we take it directly from (Shenkar & Wolf, 2022)

Distribution shift procedure : We consider two distribution shifts in the Pima Indians Diabetes dataset (Smith et al., 1988): (1) BMI Shift, where the training set includes patients with BMI < 30, and evaluation is performed on held-out sets with BMI < 30 and BMI > 30; and (2) Age Shift, where

the training set consists of patients younger than 29 (median), with evaluation on held-out sets of age ≤ 29 and age > 29.

Out-of-distribution Detection Procedure : Datasets are partitioned into training and test sets, where the training set consists exclusively of normal samples, while the test set includes both normal and anomalous samples. To account for variance and improve generalization, we leverage the *bagging effect* by averaging scores obtained from multiple feature permutations. While this approach proves particularly beneficial for small d and very small n, it comes at the cost of increased computational overhead. (Shenkar & Wolf, 2022) generate multiple feature permutations, with the number of permutations, denoted P, given by:

$$P = \min\left(\left\lfloor \frac{100}{\log(n) + d} \right\rfloor + 1, 2\right)$$

where n represents the number of training samples and d denotes the number of features.

I Auxiliary Experiments

I.1 Additional Bayesian Decision Tree Baselines

As it was hard to reproduce some of the algorithms below, namely ones in Cochrane et al. (2024), we test ours against them in the setting of Cochrane et al. (2024). We elaborate on that below. The experimental setup and DT-GFN details are identical to those in §5.1, unless otherwise stated in Table 9

Table 9: **Dataset characteristics.**

Dataset	n	d	Train	Test	Split Seeds	Experiment
Iris (Fisher, 1936)	150	4	105	45	[123456789]	DT
Wine (Aeberhard et al., 1992)	178	13	124	54	[123456789]	DT
Breast Cancer$^{(O)}$ (Wolberg et al., 1995)	699	9	489	210	[123456789]	DT
Raisin (Güvenir & Erel, 2017)	900	7	630	270	[123456789]	DT

In Table 10, using the same experimental setup as Cochrane et al. (2024), we observe that samples from a single tree generated by DT-GFN perform comparably—or often significantly better—than state-of-the-art generalization baselines.

Table 10: **Benchmarking** DT-GFN **with Bayesian decision tree baselines in** Cochrane et al. (2024), in the same setting as the latter.

Algorithm ↓ Dataset →	Iris	Wine	Breast Cancer$^{(O)}$	Raisin
BCART (Chipman et al., 1998)	$0.908_{\pm 0.022}$	$0.916_{\pm 0.046}$	$0.939_{\pm 0.014}$	$0.843_{\pm 0.010}$
SMC (Lakshminarayanan et al., 2013)	$0.909_{\pm 0.022}$	$0.978_{\pm 0.022}$	$0.924_{\pm 0.010}$	$0.842_{\pm 0.010}$
WU (Wu et al., 2007)	N/A	N/A	$0.922_{\pm 0.017}$	$0.843_{\pm 0.012}$
HMC-DF (Cochrane et al., 2023)	$0.906_{\pm 0.026}$	$0.950_{\pm 0.039}$	$0.940_{\pm 0.010}$	$0.847_{\pm 0.004}$
HMC-DFI (Cochrane et al., 2023)	$0.917_{\pm 0.023}$	$0.948_{\pm 0.022}$	$0.952_{\pm 0.007}$	$0.838_{\pm 0.007}$
DCC-TREE (Cochrane et al., 2024)	$0.911_{\pm 1.2e\text{-}16}$	$0.958_{\pm 0.02}$	$0.952_{\pm 0.004}$	$0.844_{\pm 0.002}$
DT-GFN (ours)	**0.977**	**0.981**	**0.98**	**0.856**

I.2 In-distribution/Out-of-distribution Plots with Ensemble Size Ablations

We more clearly visualize ablations in ensemble size for tree-based methods from Fig. 2 in Fig. 5.

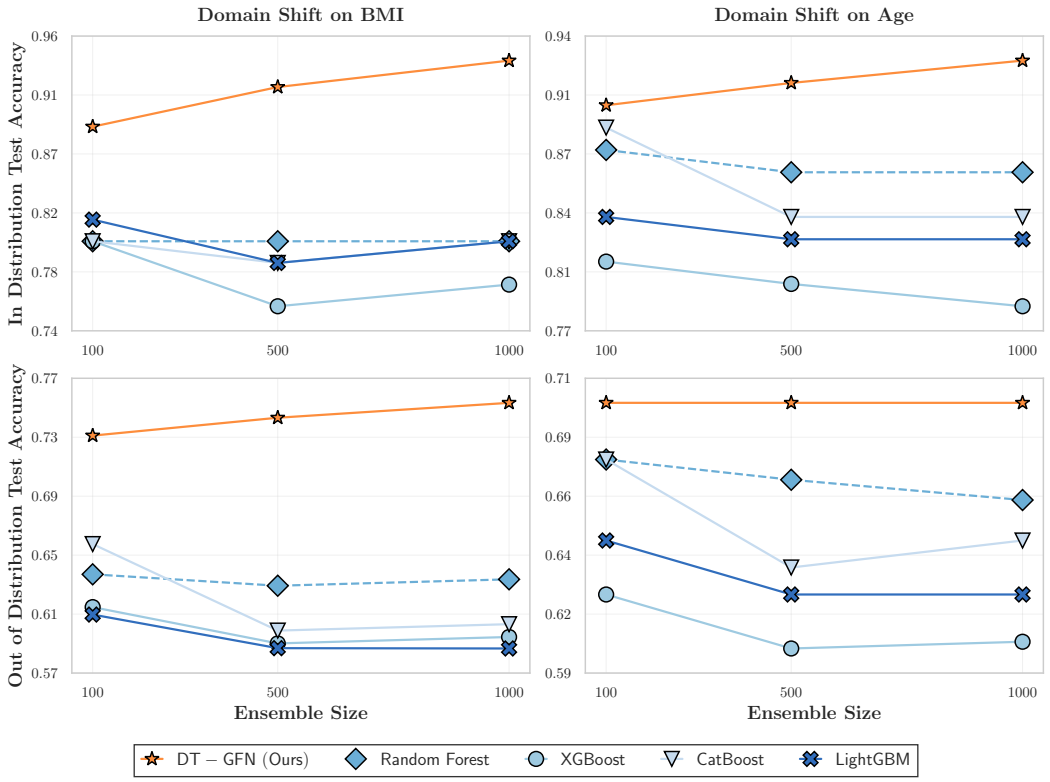

Figure 5: **Systematic increase in generalization accuracy both in-distribution and out-of-distribution in the ensemble size for tree-based methods.**

J Baseline Details

We list important intuitions and/or reproducibility guidelines for some of the baselines we compare with, and further provide code for baselines in our code base at anonymous.4open.science.

Sequential Monte Carlo (SMC) Trees (Lakshminarayanan et al., 2013). SMC Trees is a Bayesian decision tree method that employs a Sequential Monte Carlo (SMC) approach to optimize tree construction. The method maintains a population of particles, each representing a candidate split defined by a feature and threshold, and iteratively updates them based on a Dirichlet likelihood and a prior on tree complexity. The model's hyperparameters—α, which controls the Dirichlet prior, and $n_{\text{particles}}$, which regulates exploration—determine the trade-off between search diversity and convergence speed. The maximum depth is set to 5.

MAPTree (Sullivan et al., 2024). MAPTree is a Bayesian decision tree method that constructs decision trees by performing maximum a posteriori (MAP) inference over a posterior distribution of tree structures and parameters. This approach balances model complexity and predictive performance by optimizing the posterior distribution of both tree structures and parameters. As mentioned in the original paper, we impose a 300-second time limit and systematically vary the number of tree expansions (*e.g.*, 10, 100, 1000, 10000) to explore the trade-off between search depth and runtime. Additionally, we select (α, β, ρ) values to influence the prior distribution and regularization strength in the Bayesian framework.

(α^*, β^*)-Tsallis Entropy (Balcan & Sharma, 2024). Tsallis entropy, as introduced in Appendix C, is a generalization of Shannon entropy used in information theory, parametrized by α^* and β^*, which control the degree of non-extensivity. In this context, the Tsallis entropy-based decision tree method introduces these entropy measures into the tree construction process, influencing the selection of splits. The method optimizes for both diversity and accuracy in the node partitions, making it particularly suited for data with complex or hierarchical structures. The model's hyperparameters (α^*, β^*) are selected to balance between exploration of the feature space and exploitation of high-accuracy splits.

Best (α^*, β^*) **Search (Balcan & Sharma, 2024).** We follow the grid search guidelines in Balcan & Sharma (2024) to find the best (α^*, β^*) combination given training data. In particular, we use the following (α^*, β^*) tuples for each of our datasets.

Dataset ↓ Seed →	**1**	**2**	**3**	**4**	**5**
Iris	$(0.5, 1)$	$(0.5, 1)$	$(0.5, 2)$	$(0.5, 4)$	$(2, 1)$
Wine	$(1.1, 7)$	$(0.5, 1)$	$(0.5, 1)$	$(0.5, 1)$	$(0.5, 1)$
Breast Cancer Diagnostic	$(2, 1)$	$(2, 2)$	$(7, 1)$	$(2, 2)$	$(6, 1)$
Raisin	$(2, 2)$	$(2, 1)$	$(2, 1)$	$(2, 1)$	$(4, 1)$

Table 11: **Best** (α^*, β^*) (Balcan & Sharma, 2024) **for each considered dataset and split seed in §5.1.**

DPDT-4 (Hector Kohler, 2025). DPDT-4 is a parameterized decision tree algorithm that introduces a flexible partitioning strategy to create deeper and more expressive trees. The "4" in DPDT-4 refers to the maximum depth of the trees, which is designed to balance between model complexity and interpretability. This method focuses on minimizing the depth of the tree while maintaining high predictive accuracy, making it computationally efficient for large datasets.